

# vCloud SDK for .NET Developer's Guide

vCloud Director 1.5

This document supports the version of each product listed and supports all subsequent versions until the document is replaced by a new edition. To check for more recent editions of this document, see <http://www.vmware.com/support/pubs>.

EN-000613-00

**vmware**<sup>®</sup>

You can find the most up-to-date technical documentation on the VMware Web site at:

<http://www.vmware.com/support/>

The VMware Web site also provides the latest product updates.

If you have comments about this documentation, submit your feedback to:

[docfeedback@vmware.com](mailto:docfeedback@vmware.com)

Copyright © 2010, 2011 VMware, Inc. All rights reserved. This product is protected by U.S. and international copyright and intellectual property laws. VMware products are covered by one or more patents listed at <http://www.vmware.com/go/patents>.

VMware is a registered trademark or trademark of VMware, Inc. in the United States and/or other jurisdictions. All other marks and names mentioned herein may be trademarks of their respective companies.

**VMware, Inc.**  
3401 Hillview Ave.  
Palo Alto, CA 94304  
[www.vmware.com](http://www.vmware.com)

# Contents

vCloud SDK for .NET Developer's Guide	5
<b>1 About the VMware vCloud API</b>	<b>7</b>
Object Taxonomy	8
Objects, References, and Representations	9
Links and Link Relations	10
Client Workflow Overview	13
About the Schema Reference Downloadable Archive	15
<b>2 Setting Up for .NET Development</b>	<b>19</b>
Download and Install vCloud SDK for .NET	20
Using the vCloud SDK for .NET With API Version 1.0 Clients	20
<b>3 Overview of vCloud SDK for .NET Libraries and Examples</b>	<b>23</b>
Build the Example Programs	24
Run the HellovCloud Example	24
Understanding the HellovCloud Example	25
Index	29



# vCloud SDK for .NET Developer's Guide

---

The *vCloud SDK for .NET Developer's Guide* provides information about the .NET SDK for version 1.5 of the vCloud API.

VMware provides APIs and SDKs for various applications and goals. This guide provides information about the vCloud API for developers who want to create RESTful clients of VMware vCloud Director.

## Revision History

The *vCloud SDK for .NET Developer's Guide* is revised with each release of the product or when necessary. A revised version can contain minor or major changes.

**Table 1.** Revision History

Revision Date	Description
15SEP11	API Version 1.5
30AUG10	API Version 1.0

## Intended Audience

This information is intended for software developers who are building VMware Ready Cloud Services, including interactive clients of VMware vCloud Director. This information is written for software developers who are familiar with the C# programming language and .NET framework, representational State Transfer (REST) and RESTful programming conventions, the Open Virtualization Format Specification, and VMware Virtual machine technology.



# About the VMware vCloud API

---

The VMware vCloud API provides support for developers who are building interactive clients of VMware vCloud Director using a RESTful application development style.

vCloud API clients and vCloud Director servers communicate over HTTP, exchanging representations of vCloud objects. These representations take the form of XML elements. You use HTTP GET requests to retrieve the current representation of an object, HTTP POST and PUT requests to create or modify an object, and HTTP DELETE requests to delete an object.

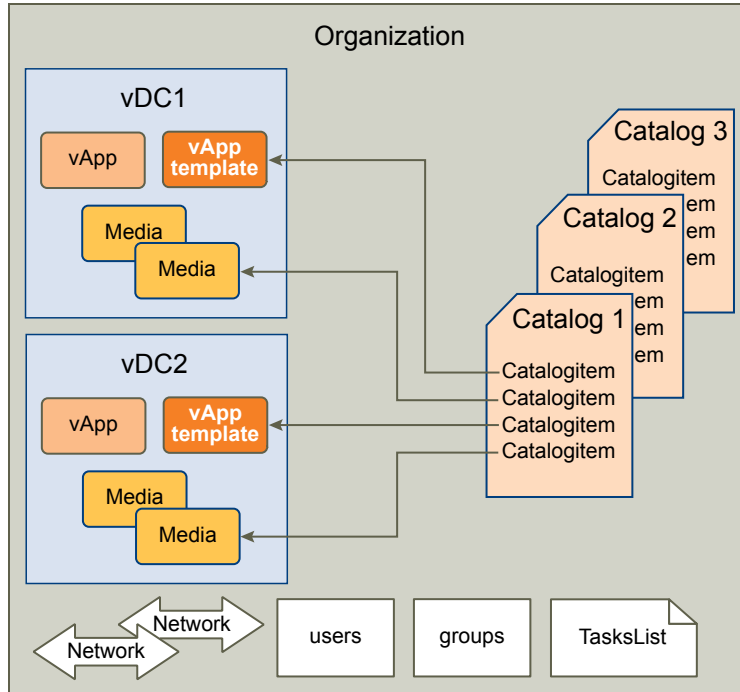
This chapter includes the following topics:

- [“Object Taxonomy,”](#) on page 8
- [“Objects, References, and Representations,”](#) on page 9
- [“Links and Link Relations,”](#) on page 10
- [“Client Workflow Overview,”](#) on page 13
- [“About the Schema Reference Downloadable Archive,”](#) on page 15

## Object Taxonomy

The vCloud API defines a set of objects common to cloud computing environments. An understanding of these objects, their properties, and their relationships is essential to using the vCloud API.

**Figure 1-1.** vCloud API Object Taxonomy



vCloud API objects have the following high-level properties:

### Organizations

A cloud can contain one or more organizations. Each organization is a unit of administration for a collection of users, groups, and computing resources. Users authenticate at the organization level, supplying credentials established when the user was created or imported.

### Users and Groups

An organization can contain an arbitrary number of users and groups. Users can be created by the organization administrator or imported from an LDAP directory service. Groups must be imported from the directory service. Permissions within an organization are controlled through the assignment of rights and roles to users and groups.

### Catalogs

Catalogs contain references to virtual systems and media images. A catalog can be shared to make it visible to other members of an organization, and can be published to make it visible to administrators in other organizations. A system administrator specifies which organizations can publish catalogs, and an organization administrator controls access to catalogs by organization members.

### Networks

An organization can be provisioned with one or more networks. These organization networks can be configured to provide services such as DHCP, NAT, VPN, and firewalls.

**Virtual Datacenters**

A virtual datacenter (vDC) is a deployment environment for virtual systems and an allocation mechanism for resources such as networks, storage, CPU, and memory. In a vDC, computing resources are fully virtualized, and can be allocated based on demand, service level requirements, or a combination of the two.

**Virtual Systems and Media Images**

Virtual systems and media images are stored in a vDC and can be included in a catalog. Media images are stored in their native representation (ISO or floppy). Virtual systems are initially stored as templates, using an open standard format (OVF 1.0). These templates can be retrieved from catalogs and transformed into virtual systems, called vApps, through a process called instantiation, which binds a template's abstract resource requirements to resources available in a vDC. A vApp contains one or more individual virtual machines (Vm elements), along with parameters that define operational details:

- How the contained virtual machines are connected to each other and to external networks.
- The order in which individual virtual machines are powered on or off.
- End-user license agreement terms for each virtual machine.
- Deployment lease terms, typically inherited from the containing organization, that constrain the consumption of vDC resources by the vApp.
- Access control information specifying which users and groups can perform operations such as deploy, power on, modify, and suspend on the vApp and the virtual machines that it contains.

**Tasks**

Asynchronous operations that members of an organization initiate are tracked by task objects, which are kept on the organization's tasks list.

## Objects, References, and Representations

The vCloud API represents objects as XML documents in which object properties are encoded as elements and attributes with typed values and an explicit object hierarchy defined by an XML schema.

XML representations of first-class vCloud API objects, such as the objects in [Figure 1-1](#), include these attributes.

<b>id</b>	The object identifier, expressed in URN format. The value of the <code>id</code> attribute uniquely identifies the object, persists for the life of the object, and is never reused. The <code>id</code> attribute value is intended to provide a context-free identifier that can be used with the vCloud API <code>entityResolver</code> and is also suitable for use by clients that need to access the object using a different API.
<b>type</b>	The object type, specified as a MIME content type.
<b>href</b>	An object reference, expressed in URL format. Because this URL includes the object identifier portion of the <code>id</code> attribute value, it uniquely identifies the object, persists for the life of the object, and is never reused. The value of the <code>href</code> attribute is a reference to a view of the object, and can be used to access a representation of the object that is valid in a particular context. Although URLs have a well-known syntax and a well-understood interpretation, a client should treat each <code>href</code> as an opaque string. The rules that govern how the server constructs <code>href</code> strings might change in future releases.

## Example: Object id, type, and href Attributes

This XML fragment, extracted from the representation of a vApp, shows its `id`, `type`, and `href` attributes.

```
<VApp
  ...
  id="urn:vcloud:vapp:490af534-1491-452e-8ed6-a5eb54447dac"
  type="application/vnd.vmware.vcloud.vApp+xml"
  href="https://vcloud.example.com/api/vApp/vapp-490af534-1491-452e-8ed6-a5eb54447dac"
  ... >
  ...
</VApp>
```

## Links and Link Relations

The vCloud API makes extensive use of `Link` elements to provide references to objects and the actions that they support. These elements are the primary mechanism by which a server tells a client how to access and operate on an object.

The server creates `Link` elements in a response body. They are read-only at the client. If a request body includes a `Link` element, the server ignores it.

### Attributes of a Link Element

In the XML representation of a vCloud object, each `Link` element has the following form:

```
<Link rel="relationship"
  type="application/vnd.vmware.vcloud.object_type+xml"
  href="URL"
  name="string"/>
```

Attribute values in a `Link` element supply the following information:

<b>rel</b>	Defines the relationship of the link to the object that contains it. A relationship can be the name of an operation on the object, a reference to a contained or containing object, or a reference to an alternate representation of the object. The relationship value implies the HTTP verb to use when you use the link's <code>href</code> value as a request URL.
<b>type</b>	The object type, specified as a MIME content type, of the object that the link references. This attribute is present only for links to objects. It is not present for links to actions.
<b>href</b>	An object reference, expressed in URL format. Because this URL includes the object identifier portion of the <code>id</code> attribute value, it uniquely identifies the object, persists for the life of the object, and is never reused. The value of the <code>href</code> attribute is a reference to a view of the object, and can be used to access a representation of the object that is valid in a particular context. Although URLs have a well-known syntax and a well-understood interpretation, a client should treat each <code>href</code> as an opaque string. The rules that govern how the server constructs <code>href</code> strings might change in future releases.
<b>name</b>	The name of the referenced object, taken from the value of that object's <code>name</code> attribute. Action links do not include a <code>name</code> attribute.

**Table 1-1.** Link Relationships and HTTP Request Types

<b>rel Attribute Value</b>	<b>Action or Relationship Description</b>	<b>Implied HTTP Verb</b>
add	Add an item to this container.	POST
alternate	References an alternate representation of this object.	GET
catalogItem	References the <code>CatalogItem</code> object that refers to this object.	GET
collaboration:abort	Abort this blocking task.	POST
collaboration:fail	Fail this blocking task.	POST
collaboration:resume	Resume this blocking task.	POST
consolidate	Consolidate this virtual machine.	POST
controlAccess	Apply access controls.	POST
copy	Reserved, unimplemented.	N/A
deploy	Deploy this vApp.	POST
disable	Disable this object.	POST
discardState	Discard the suspended state of this virtual machine.	POST
down	References an object contained by this object.	GET
download:alternate	Reserved, unimplemented.	N/A
download:default	References the default location from which this file can be downloaded.	GET
edit	Modify this object.	PUT
enable	Enable this object.	POST
firstPage	Reference to the first page of a paginated response.	GET
installVmwareTools	Install VMware Tools on this virtual machine.	POST
lastPage	Reference to the last page of a paginated response.	GET
media:ejectMedia	Eject virtual media from a virtual device.	POST
media:insertMedia	Insert virtual media into a virtual device.	POST
move	Reserved, unimplemented.	N/A
nextPage	Reference to the next page of a paginated response.	GET
ova	Reserved, unimplemented	N/A
ovf	References the OVF descriptor of this vApp template.	GET
power:powerOff	Power off this vApp or virtual machine.	POST
power:powerOn	Power on this vApp or virtual machine.	POST
power:reboot	Reboot this vApp or virtual machine.	POST
power:reset	Reset this vApp or virtual machine.	POST

**Table 1-1.** Link Relationships and HTTP Request Types (Continued)

<b>rel Attribute Value</b>	<b>Action or Relationship Description</b>	<b>Implied HTTP Verb</b>
power:shutdown	Shut down this vApp or virtual machine.	POST
power:suspend	Suspend this vApp or virtual machine.	POST
previousPage	Reference to the previous page of a paginated response.	GET
publish	Publish this catalog.	POST
recompose	Recompose this vApp.	POST
reconnect	Reconnect this vCenter Server to this cloud.	POST
register	Register a VCenter Server to this cloud.	POST
reject	Reject this request.	POST
relocate	Relocate this virtual machine.	POST
remove	Remove this object.	DELETE
repair	Repair this ESX/ESXi host.	POST
screen:acquireTicket	Retrieve a screen ticket for this virtual machine.	GET
screen:thumbnail	Retrieve a thumbnail view of the screen of this virtual machine.	GET
task:cancel	Cancel this task.	POST
blockingTask	A list of pending blocking task requests in this cloud.	GET
taskOwner	Reference to the owner of a task	GET
taskParams	Reference to the request parameters of a task	GET
taskRequest	Reference to the request associated with a task	GET
undeploy	Undeploy this vApp.	POST
unlock	Unlock a user account	POST
unregister	Unregister this vCenter Server.	POST
up	References an object that contains this object.	GET
updateProgress	Request an update of this task's progress.	POST
upgrade	Upgrade this ESX/ESXi host.	POST
upload:alternate	Reserved, unimplemented.	N/A
upload:default	References the default location to which this object can be uploaded.	PUT

## Client Workflow Overview

vCloud API clients implement a RESTful workflow, making HTTP requests to the server and retrieving the information they need from the server's responses.

### About RESTful Workflows

REST, an acronym for Representational State Transfer, describes an architectural style characteristic of programs that rely on the inherent properties of hypermedia to create and modify the state of an object whose serialized representation is accessible at a URL.

If a URL of such an object is known to a client, the client can use an HTTP GET request to retrieve the representation of the object. In the vCloud API, this representation is an XML document. In a RESTful workflow, documents that represent of object state are passed back and forth between a client and a service with the explicit assumption that neither party need know anything about an object other than what is presented in a single request or response. The URLs at which these documents are available often persist beyond the lifetime of the request or response that includes them. The other content of the documents is nominally valid until the expiration date noted in the HTTP Expires header.

### vCloud REST API Workflows

Application programs written to a REST API use HTTP requests that are often executed by a script or other higher-level language to make remote procedure calls that create, retrieve, update, or delete objects that the API defines. In the vCloud REST API, these objects are defined by a collection of XML schemas. The operations themselves are HTTP requests, and so are generic to all HTTP clients.

To write a RESTful client, you must understand only the HTTP protocol and the semantics of XML, the transfer format that the vCloud API uses. To use the vCloud API effectively in such a client, you need to know only a few things:

- What is the set of objects that the API supports, and what do they represent. For example, what is a vDC and how does it relate to an organization or catalog?
- How does the API represents these objects. For example, what does the XML schema for an Org look like? What do the individual elements and attributes represent?
- How does the client refer to an object on which it wants to operate. For example, where are the links to objects in a vDC? How does a client obtain and use them?

You can find this information in the vCloud API XML schemas. The XML elements, attributes, and composition rules defined in these schemas and represent the data structures of objects in the cloud. A client can read an object by making an HTTP GET request to the object's URL. A client can create or modify an object with an HTTP PUT or POST request that includes a new or changed XML body document for the object. A client can usually delete an object with an HTTP DELETE request.

The vCloud API schema reference includes detailed information about the XML representations of all vCloud API objects and examples of HTTP requests that operate on those objects. See [“About the Schema Reference Downloadable Archive,”](#) on page 15.

### RESTful Workflow Patterns

All RESTful workflows follow a common pattern.

- 1 Make an HTTP request, typically GET, PUT, POST, or DELETE. The target of this request is either a well-known URL such as a the vCloud API versions URL, or a URL obtained from the response to a previous request. For example, a GET request to an organization URL returns links to catalog and vDC objects that the organization contains.

- 2 Examine the response, which always includes an HTTP response code and usually includes a body. In the vCloud API, a response body is an XML representation of an object, including elements and attributes that represent object properties, links that implement operations on the object or provide references to contained or containing objects and, if the object is being created or modified, an embedded task object that tracks the progress of the creation or modification. The response also includes an HTTP response code, which indicates whether the request succeeded or failed, and might be accompanied by a URL that points to a location from which you can retrieve additional information.

These operations can repeat, in this order, for as long as necessary.

## vCloud API REST Requests

To retrieve object representations, clients make HTTP requests to object references. The server supplies these references as href attribute values in responses to GET requests.

Every cloud has a well-known URL from which an unauthenticated user can retrieve a list of vCloud API versions that the server supports. Each version has its own login URL. A system administrator can use that URL to authenticate to the cloud by logging in to the System organization. An authenticated user can discover other vCloud API URLs by making GET requests to URLs retrieved from the login response, and the URLs contained in responses to those requests. See .

Requests are typically categorized in terms of the type of requested operation: create, retrieve, update, and delete. This sequence of verbs is often abbreviated with the acronym CRUD.

**Table 1-2.** CRUD Operations Summary

Operation Type	HTTP Verb	Operation Summary
Create	POST	Creates a new object.
Retrieve	GET	Retrieves the representation of an existing object.
Update	PUT	Modifies an existing object.
Delete	DELETE	Deletes an existing object.

## vCloud API REST Responses

All responses include an HTTP status code and, unless the status code is 204 (No Content), a Content-Type header. Response content depends on the request. Some responses include a document body, some include only a URL, and some are empty.

A vCloud API client can expect a subset of HTTP status codes in a response.

**Table 1-3.** HTTP Status Codes that the vCloud API Returns

Status Code	Status Description
200 OK	The request is valid and was completed. The response includes a document body.
201 Created	The request is valid. The requested object was created and can be found at the URL specified in the Location header.
202 Accepted	The request is valid and a task was created to handle it. This response is usually accompanied by a Task element.
204 No Content	The request is valid and was completed. The response does not include a body.
303 See Other	The response to the request can be found at the URL specified in the Location header.
400 Bad Request	The request body is malformed, incomplete, or otherwise invalid.

**Table 1-3.** HTTP Status Codes that the vCloud API Returns (Continued)

Status Code	Status Description
401 Unauthorized	An authorization header was expected but not found.
403 Forbidden	The requesting user does not have adequate privileges to access one or more objects specified in the request.
404 Not Found	One or more objects specified in the request could not be found in the specified container.
405 Method Not Allowed	The HTTP method specified in the request is not supported for this object.
500 Internal Server Error	The request was received but could not be completed because of an internal error at the server.
501 Not Implemented	The server does not implement the request.
503 Service Unavailable	One or more services needed to complete the request are not available on the server.

## About the Schema Reference Downloadable Archive

XML schema reference documentation in HTML format for the vCloud API is available as a downloadable archive. This archive also includes the schema definition files, and examples XML representations of vCloud API objects.

To use the reference documentation:

- 1 Download the compressed archive from <http://www.vmware.com/support/vcd/doc/rest-api-doc-1.5-html.zip>
- 2 Uncompress the archive into any convenient folder.
- 3 In the folder, open the file `index.html` in a browser.

## How the Schema Reference Documentation is Organized

The schema reference documentation is organized to reflect the division of the vCloud API into user, administrator, and extension categories. Within each category, you can open a list of elements, types that the elements extend, and operations that create, retrieve, update, or delete the objects that the elements represent.

### **User Operations, Elements, and Types**

These operations are performed by all users who have permission to log into an organization. User elements and user types represent the objects that these operations manipulate.

### **Administrator Operations, Elements, and Types**

These operations are performed by organization administrators or system administrators. Administrator elements and types represent the objects that these operations manipulate.

### **Extension Operations, Elements, and Types**

These operations are performed by system administrators who need access to vSphere platform objects from the vCloud API. Extension elements and types represent the objects that these operations manipulate.

## Searching In a Category

You can enter a search string in the Quick Index text box to search the lists of operations, elements, and types in any category.

- In an **Operations** list, you can search for the following items:
  - All or part of the name of the object on which you want to operate. The search returns a list of all of the operations that are possible on that object. For example, selecting User Operations and typing **vApp** in the Quick Index text box returns a list of all of the requests that operate on a vApp object.
  - The name of an action to perform. For example, selecting User Operations and typing **power** in the Quick Index text box returns a list of all the requests that change the power state of a vApp.
  - An HTTP verb (GET, PUT, POST, DELETE) to view a list of all the requests that use that verb. For example, selecting User Operations and typing **PUT** in the Quick Index text box returns a list of all of the requests that update an object.
- In an **Elements** or **Types** list, type all or part of the element or type name.

Search terms are not case-sensitive.

## Operation Summary Syntax

Operations consist of an HTTP verb and a request URL. The reference documentation represents the verb and the URL using the following syntax:

```
HTTP_VERB /object_type/{id}[/action/action_name]
```

In this syntax, the initial / character is assumed to follow a site-specific API URL, such as `https://vcloud.example.com/api`. The following strings represent variables in the remainder of the URL:

<b>HTTP_VERB</b>	The HTTP verb used to request the operation.
<b>object_type</b>	An abbreviation of the MIME type of the object referenced by the operation. This abbreviation is constructed from the final component of the object's media type, between the . and the +xml designation. For example, for an object whose media type is <code>application/vnd.vmware.vcloud.catalogItem+xml</code> , the <i>object_type</i> is shown as <code>catalogItem</code> .
<b>{id}</b>	The unique identifier of the object of the operation.
<b>action_name</b>	The name of an action. Required only when the operation request URL includes the string <code>/action/</code> .

## Element and Type Reference Pages

For each element or complex type, the reference documentation provides a page that lists the following items:

<b>Element</b>	The name of the element.
<b>Type</b>	The name of the type that the element extends.
<b>Namespace</b>	The XML namespace in which this element or type name is defined.
<b>Description</b>	A description of the purpose and contents of the element or type.
<b>Since</b>	The vCloud API version in which this element or type first appeared.
<b>Schema</b>	The name of the XML schema definition file in which this element or type is defined. Click to open the file in your browser, or right-click to download it.

<b>Media Type</b>	The MIME type associated with this element or type.																		
<b>Extends</b>	The base type from which this element is derived.																		
<b>XML Representation</b>	The XML representation of the element or type. Names of contained elements are links to the reference pages for those elements.																		
<b>Attributes</b>	A table listing the following properties of each attribute of the element or type: <table> <tr> <td><b>Attribute</b></td> <td>The name of the attribute.</td> </tr> <tr> <td><b>Type</b></td> <td>The primitive XML type of the attribute.</td> </tr> <tr> <td><b>Required</b></td> <td>Yes for attributes that are required. No for attributes that are optional.</td> </tr> <tr> <td><b>Modifiable</b></td> <td>A value of <code>always</code> means that a client request can modify the value of this attribute. A value of <code>create</code> means that this attribute can be set or modified only as part of object creation. A value of <code>none</code> means that this attribute is read-only.</td> </tr> <tr> <td><b>Since</b></td> <td>The vCloud API version in which this attribute first appeared.</td> </tr> <tr> <td><b>Description</b></td> <td>A description of the purpose and contents of the attribute.</td> </tr> </table>	<b>Attribute</b>	The name of the attribute.	<b>Type</b>	The primitive XML type of the attribute.	<b>Required</b>	Yes for attributes that are required. No for attributes that are optional.	<b>Modifiable</b>	A value of <code>always</code> means that a client request can modify the value of this attribute. A value of <code>create</code> means that this attribute can be set or modified only as part of object creation. A value of <code>none</code> means that this attribute is read-only.	<b>Since</b>	The vCloud API version in which this attribute first appeared.	<b>Description</b>	A description of the purpose and contents of the attribute.						
<b>Attribute</b>	The name of the attribute.																		
<b>Type</b>	The primitive XML type of the attribute.																		
<b>Required</b>	Yes for attributes that are required. No for attributes that are optional.																		
<b>Modifiable</b>	A value of <code>always</code> means that a client request can modify the value of this attribute. A value of <code>create</code> means that this attribute can be set or modified only as part of object creation. A value of <code>none</code> means that this attribute is read-only.																		
<b>Since</b>	The vCloud API version in which this attribute first appeared.																		
<b>Description</b>	A description of the purpose and contents of the attribute.																		
<b>Elements</b>	A table listing the following properties of each element defined in the type: <table> <tr> <td><b>Element</b></td> <td>The name of the element.</td> </tr> <tr> <td><b>Type</b></td> <td>A link to the definition of the complex type that the element is based on.</td> </tr> <tr> <td><b>Occurrence</b></td> <td>The occurrence constraint for the element. The constraint can be one of the following expressions: <table> <tr> <td><b>0..*</b></td> <td>Optional. Can occur zero or more times.</td> </tr> <tr> <td><b>0..1</b></td> <td>Optional. Can occur at most once.</td> </tr> <tr> <td><b>1</b></td> <td>Required. Must occur exactly once.</td> </tr> </table> </td> </tr> <tr> <td><b>Modifiable</b></td> <td>A value of <code>always</code> means that a client request can modify the contents of this element. A value of <code>create</code> means that element contents can be set or modified only as part of object creation. A value of <code>none</code> means that this element is read-only.</td> </tr> <tr> <td><b>Since</b></td> <td>The vCloud API version in which this element first appeared.</td> </tr> <tr> <td><b>Description</b></td> <td>A description of the purpose and contents of the element.</td> </tr> </table>	<b>Element</b>	The name of the element.	<b>Type</b>	A link to the definition of the complex type that the element is based on.	<b>Occurrence</b>	The occurrence constraint for the element. The constraint can be one of the following expressions: <table> <tr> <td><b>0..*</b></td> <td>Optional. Can occur zero or more times.</td> </tr> <tr> <td><b>0..1</b></td> <td>Optional. Can occur at most once.</td> </tr> <tr> <td><b>1</b></td> <td>Required. Must occur exactly once.</td> </tr> </table>	<b>0..*</b>	Optional. Can occur zero or more times.	<b>0..1</b>	Optional. Can occur at most once.	<b>1</b>	Required. Must occur exactly once.	<b>Modifiable</b>	A value of <code>always</code> means that a client request can modify the contents of this element. A value of <code>create</code> means that element contents can be set or modified only as part of object creation. A value of <code>none</code> means that this element is read-only.	<b>Since</b>	The vCloud API version in which this element first appeared.	<b>Description</b>	A description of the purpose and contents of the element.
<b>Element</b>	The name of the element.																		
<b>Type</b>	A link to the definition of the complex type that the element is based on.																		
<b>Occurrence</b>	The occurrence constraint for the element. The constraint can be one of the following expressions: <table> <tr> <td><b>0..*</b></td> <td>Optional. Can occur zero or more times.</td> </tr> <tr> <td><b>0..1</b></td> <td>Optional. Can occur at most once.</td> </tr> <tr> <td><b>1</b></td> <td>Required. Must occur exactly once.</td> </tr> </table>	<b>0..*</b>	Optional. Can occur zero or more times.	<b>0..1</b>	Optional. Can occur at most once.	<b>1</b>	Required. Must occur exactly once.												
<b>0..*</b>	Optional. Can occur zero or more times.																		
<b>0..1</b>	Optional. Can occur at most once.																		
<b>1</b>	Required. Must occur exactly once.																		
<b>Modifiable</b>	A value of <code>always</code> means that a client request can modify the contents of this element. A value of <code>create</code> means that element contents can be set or modified only as part of object creation. A value of <code>none</code> means that this element is read-only.																		
<b>Since</b>	The vCloud API version in which this element first appeared.																		
<b>Description</b>	A description of the purpose and contents of the element.																		
<b>Operations</b>	A summary of the operations permitted on the element. Operations are categorized by request type; one of create, retrieve, update, and delete. This sequence of verbs is often abbreviated with the acronym CRUD.																		

## Schema Definition Files

XML schema definition files (\*.xsd) are included in the etc folder of schema reference downloadable archive. This folder contains several subfolders:

- 1.0** Schema definition files for vCloud API version 1.0.
- 1.5** Schema definition files for vCloud API version 1.5.
- schemas** Additional schema definition files that are version-independent or from external sources such as DMTF.

# Setting Up for .NET Development

---

To use the vCloud SDK for .NET, you need Microsoft Visual Studio and the .NET framework.

## Prerequisites for .NET Development

Verify that you have the following software installed on the development host:

- Microsoft Visual Studio 2008 or later.
- Microsoft .NET Framework 3.5 or later.
- Additional DLL files, as documented in the README file in the download.

In addition, consider the following items:

- The vCloud SDK for .NET reference documentation provides information about the vCloud API XML schemas, which define the objects and operations that the SDK supports. Familiarity with the details of the underlying objects and operations, as described in the *vCloud API Programming Guide*, can help you understand the structure of vCloud API objects, and how the methods in this SDK operate on those objects.
- Before you can run the examples, you must use the vCloud Director Web console or the vCloud API to create an organization, catalog, and vDC that the samples can use. The organization must have a user account with rights to run the samples. The predefined `CatalogAuthor` role should provide all the necessary rights. For more information about roles and rights, see the *VMware vCloud Director Administrator's Guide*.
- Several of the sample programs, including `HellovCloud`, require you to have an OVF package available on the client host. This package must be uncompressed, and must specify a single vmdk file. For more information about OVF, see the *vCloud API Programming Guide*.

This chapter includes the following topics:

- [“Download and Install vCloud SDK for .NET,”](#) on page 20
- [“Using the vCloud SDK for .NET With API Version 1.0 Clients,”](#) on page 20

## Download and Install vCloud SDK for .NET

You can download the vCloud SDK for .NET from the VMware Web site. The SDK is distributed as a compressed archive named `VMware-vCloudDirector-.NetSDK-1.5.0-build.zip`, where *build* is the build number of the SDK.

Uncompressed, the archive requires about 40 MB of disk space. The package includes DLL files and the following folders:

<b>Docs</b>	vCloud SDK for .NET reference documentation in HTML format.
<b>Samples</b>	Example code demonstrating common use cases associated with programmatically managing virtual infrastructure.

### Procedure

- 1 In a browser, go to <http://www.vmware.com/go/vcloudsdkfordotnet>.
- 2 In the Resources area of the vCloud SDK for .NET Community page, click **Download**.
- 3 On the Download page, log in with your VMware customer credentials.
- 4 Review the license agreement and click **Yes** to accept it and continue.
- 5 On the Download page, choose a download option and click the file format to download.
- 6 When the download is complete, uncompress the download package into any convenient folder on your computer.
- 7 Import the package to Visual Studio.

### What to do next

For information about additional DLL files that you must obtain, see the README file in the download.

## Using the vCloud SDK for .NET With API Version 1.0 Clients

In the native configuration, this SDK supports version 1.5 of the vCloud API. It includes additional files that allow it to support vCloud API 1.0 clients.

To use this SDK to enable a client written for the vCloud API 1.0 to work with vCloud Director 1.5, you must replace several of the default dll files in your IDE and change some import statements in your client code.

### Procedure

- 1 In your IDE, replace `VcloudSDK.dll` with `VcloudSDK_V1_0.dll` and replace `VcloudRestSchema.dll` with `VcloudRestSchema_V1_0.dll`.
- 2 Change the import statements in your client code to import the \*\_v1\_0 versions of the `sdk`, `schema`, and extension dll files.

See “[Example: Replacing Import Statements](#),” on page 21.

## Example: Replacing Import Statements

This code fragments shows the `import` statements for a version 1.5 client commented out and replaced with `import` statements for a version 1.0 client.

```
//import com.vmware.vcloud.sdk.*  
import com.vmware.vcloud.sdk_v1_0.*  
//import com.vmware.vcloud.api.rest.schema.*  
import com.vmware.vcloud.api.rest.schema_v1_0.*  
//import com.vmware.vcloud.api.rest.schema.extension.*  
import com.vmware.vcloud.api.rest.schema_v1_0.extension.*
```



# Overview of vCloud SDK for .NET Libraries and Examples

# 3

The vCloud SDK for .NET includes libraries, examples of C# application code, and reference documentation on SDK classes and methods.

## Libraries

The SDK includes several function libraries in dll form.

**Table 3-1.** Libraries

Name	Description
RabbitMQ.Client.dll	Methods and classes for using notifications and blocking tasks
VcloudRestSchema_V1_0.dll	For use by vCloud API 1.0 clients. See <a href="#">“Using the vCloud SDK for .NET With API Version 1.0 Clients,”</a> on page 20.
VcloudRestSchema_V1_5.dll	Methods and classes for accessing the REST XML schema.
VcloudSDK_V1_0.dll	For use by vCloud API 1.0 clients. See <a href="#">“Using the vCloud SDK for .NET With API Version 1.0 Clients,”</a> on page 20.
VcloudSDK_V1_5.dll	Methods and classes for accessing REST operations.

## Examples

The SDK `samples` directory includes example programs that demonstrate how you can use the vCloud SDK for .NET to develop client applications. Users who have rights to create and modify catalog items and vApps can run user API programs.

**Table 3-2.** User API Examples

Name	Description
CatalogInventorySample	Lists <code>name</code> and <code>href</code> for all items in all catalogs in the organization.
CatalogItemCRUD	Create, retrieve, update, or delete a catalog item.
DiskCRUD	Create, retrieve, update, or delete a virtual hard disk in a <code>Vm</code> object.
HellovCloud	Implements a structured workflow through the life cycle of a vApp.
ListAllvApps	List all vApps in a vDC by name and href.

**Table 3-2.** User API Examples (Continued)

Name	Description
ThreadSample	Examples of how to implement multithreaded client applications that execute multiple requests in parallel.
VdcInventorySample.	List <i>name</i> and <i>href</i> for all vApps, vApp templates, and media images in all vDCs in the organization

Administrative examples require organization administrator privileges.

**Table 3-3.** Administrative API Samples

Name	Description
CatalogCRUD	Create, retrieve, update, or delete a catalog.
OrganizationCRUD	Create, retrieve, update, or delete an organization. Requires system administrator privileges.
OrgNetworkCRUD	Create, retrieve, update, or delete an organization network
GroupCRUD	Create, retrieve, update, or delete a Group object.
RoleCRUD	Create, retrieve, update, or delete a role.
UserCRUD	Create, retrieve, update, or delete a local user.
VdcCRUD	Create, retrieve, update, or delete a vDC
QueryAllvApps	Query all the vApps in the vCD by the System or Cloud Admin
ReceiveNotifications	Receive notifications from an AMQP Broker.

This chapter includes the following topics:

- [“Build the Example Programs,”](#) on page 24
- [“Run the HellovCloud Example,”](#) on page 24
- [“Understanding the HellovCloud Example,”](#) on page 25

## Build the Example Programs

Before you can run HellovCloud and the other example programs, you must build them in Visual Studio.

### Procedure

- 1 Open the `Samples` folder.
- 2 Double-click the `samples.sln` file.
- 3 Click **Build > Build Solution**.

## Run the HellovCloud Example

The HellovCloud example, included in the `Samples` folder of the SDK, demonstrates operations that the vCloud SDK for .NET supports.

HellovCloud demonstrates the following operations:

- Logging in to the cloud
- Uploading an OVF package to create a vApp template
- Adding the vApp template to a catalog
- Instantiating the vApp template to create a vApp

- Operating the vApp

The `HellovCloud.txt` file, also included in the `Samples` folder, contains examples of program inputs and outputs.

#### Prerequisites

Build the `HellovCloud` example. See “[Build the Example Programs](#),” on page 24.

#### Procedure

- 1 Open a console or shell in the `samples` folder.
- 2 Run the `HellovCloud fi` command..

## Example: Running HellovCloud

To run `HellovCloud`, use a command line like this example.

```
.Net HellovCloud vCloudURL user@vcloud-organization password orgName vdcName ovfFileLocation
vmdkFileLocation vmdkFileName catalogName
```

Type the following values on the command line:

<b>vCloudURL</b>	The vCloud Director server URL.
<b>user</b>	The name of a user account that can run the sample.
<b>vcloud-organization</b>	The name of the organization in which the user account exists.
<b>password</b>	The user's password.
<b>orgName</b>	The name of the organization in which the user account exists.
<b>vdcName</b>	The name of a vDC in that organization where the user can upload the OVF and deploy the vApp.
<b>ovfFileLocation</b>	The full pathname to the OVF descriptor on the local disk.
<b>vmdkFileLocation</b>	The full pathname to the vmdk file referenced in the OVF descriptor.
<b>vmdkFileName</b>	The file name of the vmdk file.
<b>catalogName</b>	The name of the catalog in which the vApp template will be catalogued.

For example:

```
.Net HellovCloud https://vcloud.example.com user@SampleOrg Pa55w0rd SampleOrg SampleVdc
C:\descriptor.ovf C:\disk.vmdk disk.vmdk SampleCatalog
```

## Understanding the HellovCloud Example

The `HellovCloud` example includes extensive comment blocks that explain how each of the steps in the example use the SDK libraries.

`HellovCloud` performs the following sequence of operations, which are typical of the workflow for provisioning and operating a vApp.

## Logging In and Getting an Organization List

Most vCloud API requests must be authenticated by a login request that supplies user credentials in the form that Basic HTTP authentication requires. MIME Base64 encoding of a string has the form *user@vcloud-organization:password*. The `VcCloudClient` class implements a login method that takes the following parameters:

<b>userName</b>	Supplied in the form <i>user@vcloud-organization</i> .
<b>password</b>	The user's password.

`HellovCloud` encapsulates this authentication protocol in its `login()` method, which returns a list of organizations to which you have access. In the typical case, this list has a single member, the organization that is supplied in the `userName` parameter.

## Getting References to the vDC and Catalog

To instantiate a vApp template and operate the resulting vApp, you need the object references, the href values, for the catalog in which the vApp template will be entered and the vDC in which the vApp will be deployed. The `Organization` class implements `findVdc()` and `findCatalogRef()` methods that return references to vDCs and catalogs. `HellovCloud` uses these methods in its `FindVdc` method.

## Uploading an OVF Package to Create a vApp Template

The `HellovCloud` command line requires that you supply the name of an OVF descriptor file and the `vmrk` file that it references. The `createUploadvAppTemplate` method uses this information to upload the OVF descriptor and `vmrk` file, create a vApp template, and return a reference to the template that other methods in the program can use.

The `createUploadvAppTemplate()` method and the methods that it calls from the vCloud SDK for .NET implement the following workflow to upload the OVF package and create a vApp template.

- 1 The client uses a POST request that specifies a name and description for the template, and a transfer format for the data.
- 2 The server returns an unresolved `VAppTemplate` element with (`status="0"`) that includes an upload URL for the OVF descriptor.
- 3 The client uses an HTTP PUT request to upload the descriptor to the upload URL.
- 4 The server reads the descriptor and modifies the `vAppTemplate` to include an upload URL for each file listed in the `References` section of the descriptor. While the server is modifying the `vAppTemplate`, the client makes periodic requests for it and examines the response for additional upload URLs. When the response contains additional upload URLs that were not present in the initial response, template construction is complete.
- 5 The client uses HTTP PUT requests to upload each of the files.

After all of the files are uploaded, and validated if a manifest is present, the server processes the uploads. When processing is complete, the server sets the value of the template's `status` attribute to 8, indicating that the template is ready for use. This status value indicates that all of the virtual machines in the template are powered off. See the *vCloud API Programming Guide*.

## Adding the vApp Template to a Catalog

After the vApp template is created, `HellovCloud` uses its `createNewCatalogItem()` method to create a `CatalogItem` object in the catalog whose name was provided on the command line. The `CatalogItem` contains the reference to the template that was returned in ["Uploading an OVF Package to Create a vApp Template,"](#) on page 26.

## Instantiating the vApp Template

After the template is added to in a catalog, you can instantiate it to create a vApp. `HelloCloud` implements a `newVAppFromTemplate()` method that has the following parameters:

<b>vAppTemplateReference</b>	A reference to the template, obtained from the catalog.
<b>Vdc</b>	A reference to the vDC in which to instantiate the template.

With these inputs, `newVAppFromTemplate()` constructs a simple `InstantiateVAppTemplateParams` request body, makes the request to the `action/instantiateVAppTemplate` URL of the vDC, and returns a helper object that contains a reference to the vApp.

## Operating the vApp

The `vapp` class includes methods that perform operations on the vApp. Most of these operations return a `Task` object that tracks the progress of the operation. `HelloCloud` uses these methods to cycle the vApp through the following states, in this order:

- 1 `vapp.deploy`, which deploys the vApp
- 2 `vapp.powerOn`, which powers on the vApp
- 3 `vapp.suspend`, which suspends the vApp
- 4 `vapp.powerOff`, which powers off the vApp
- 5 `vapp.undeploy`, which undeploys the vApp
- 6 `vapp.delete`, which deletes the vApp



# Index

## **E**

Entity, object representation in **9**  
examples, to build **24**

## **H**

HellovCloud, about **25**  
HellovCloud sample, to run **24**

## **I**

id attribute **9**

## **J**

JDK, supported versions **19**

## **L**

Link element, rel attribute **10**

## **O**

object hierarchy, diagram of **8**  
object identifiers **9**  
object references, about **9**

## **R**

requests, about **14**  
responses, about **14**

## **S**

sample programs, list of **23**  
schema files, accessing **15**  
schema reference **15**  
SDK  
    and older clients **20**  
    to download **20**

## **V**

vCloud API, and RESTful programming style **7**

## **W**

workflow **13**

## **X**

XML  
    compressed responses **14**  
    validation of **14**

